SMSP（Standard Musical String Protocol、标准音乐字符串协议）入门手册

1. 什么是SMSP

SMSP是一种将字符串解释为音乐的方法，可用于记录音乐或创作音乐。从功能上来说，SMSP的定位类似于乐谱。不过相比于传统的基于五线谱或简谱的音乐表示法，其优点在于可被键盘简单输入，同时结合了一些函数式编程和现代DAW类编曲软件的特点，使得符合SMSP标准的字符串在具备一定音乐可读性的同时又能高效地表示尽可能丰富的音乐信息。

1. 什么是SMSP编译器

以符合SMSP标准的字符串作为输入，输出相应可播放音乐文件的计算机程序均可被称为SMSP编译器。

可播放音乐文件可以是MIDI也可以是波形的形式，甚至是更底层的exe形式，但目前基于MMA的SMSP编译器只能达到前者。

1. SMSP的基本思想是什么

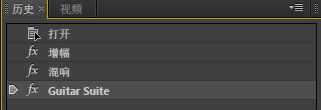
SMSP将所有的语素分为两类，一类称为音符，一类称为控制符，每个符号之间尽量用留白字符分开。其中留白字符指的是空格space，回车enter或制表符tab中的任意一个。

在有唯一理解方式的前提下，语素之间可以不留白。

音符指的是可被立即解释为具体音乐事件信息的连续字符串，可以理解为DAW软件中在钢琴卷帘上画音符条的操作。控制符指的是不被解释或不被立即解释为具体音乐事件的连续字符串，可以理解为DAW软件中的其它任何操作，比如导入文件，设置时间轴位置，切换音轨，改变配器等。

由此可以引申出SMSP中一个最重要的思想，那就是顺序解释思想。字符串必须按照从左到右和从上到下换行的顺序来解释，包括所有的控制符和音符。

你可能觉得这个很显然，但是我仍然要强调这一点，因为这里的“顺序解释”是字符串层面的，不是时间轴层面的。也就是说，SMSP记录的是你创造音乐的历史，与乐谱是完全不一样的，而更类似于DAW类软件里的这个窗口：



下面我举个例子来具体说明。考虑helloworld.txt中字母歌旋律的例子

1 1 5 5|6 6 5-|4 4 3 3|2 2 1-|

5 5 4 4|3 3 2-|5 5 4 4|3 3 2-|

1 1 5 5|6 6 5-|4 4 3 3|2 2 1-

在SMSP中，用”(小节数)”可以指定一段旋律的从第几小节开始演奏，这被称作小节控制符。如果不写这个控制符，默认是0。所以helloworld.txt的编译结果就是从头开始播放的字母歌旋律。

为了使得排版漂亮一些，我们将helloworld.txt改写为下列等价形式：

(0) 1 1 5 5|6 6 5-|4 4 3 3|2 2 1-

(4) 5 5 4 4|3 3 2-|5 5 4 4|3 3 2-

(8) 1 1 5 5|6 6 5-|4 4 3 3|2 2 1-

相当于3个4小节的音乐片段分别从第0,4,8小节开始演奏，因此结果不变。

根据顺序解释的思想，我们可以这么改写：

(8) 1 1 5 5|6 6 5-|4 4 3 3|2 2 1-

(0) 1 1 5 5|6 6 5-|4 4 3 3|2 2 1-

(4) 5 5 4 4|3 3 2-|5 5 4 4|3 3 2-

尽管可读性降低，但是结果仍然是不变的。SMSP支持用超越时间轴的概念来解释音乐。这段代码表示的是时间轴先设置到第8小节开始，添加后面对应4个小节的音符，再设置到第0小节开始，添加后面对应4个小节的音符，最后再设置到第4小节开始，添加后面对应4个小节的音符。

在更加高级的语法中，顺序解释的思想仍将发挥重要作用。比如考虑到导入文件和定义函数之后，代码可能会是下面这个样子：

…导入文件，定义函数集1….

…音符序列1….

…导入文件，定义函数集2….

…音符序列2….

那么我们一定知道音符序列1中只能用到函数集1中的函数，不能用函数集2中的函数，因为此时还没有进行到定义函数集2。而函数集2在定义过程中可以以函数集1中的函数为基础，另外音符序列2即可以用到函数集1中函数也可以用到函数集2中的函数。

1. 我要开始创作音乐了，第一步该做什么

思想与DAW类软件完全类似。试想一下你打开FL Studio这样的软件首先要做什么，首先，我们可能会导入一些素材。在SMSP中，对应的是已经集成好的字符库，可对应集成好的旋律库、和弦库和函数库等。

=>语言点：导入控制符

语法：<<文件名地址

解析：文件名地址一般是txt文件，要求UTF-8或ANSI字符编码，默认起始路径为编译器所在目录（可修改编译开关选项改变此默认路径，详见关于编译器的说明文档）。该控制符的作用是将该文件内全部的字符串插入到该控制符的对应位置，支持文件间的无限递归导入，也就是被导入的文件里也可以包含导入控制符，导入另一个文件。

举例：<<chords.txt （见幻化成风.txt）

然后我们可能要设置拍号与速度

=>语言点：方法控制符

语法：m[barbeat,bpm,base]

解析：barbeat是每小节节拍数（正整数），bpm是每分钟节拍数（正实数），均以1/4分音符为单位，然后base是基准音偏移，以十二平均律为单位（整数），值为0时表示在解释音符的音高时，0对应C4。该控制符的作用就是设置这三个全局变量的值。如果从未指定，按缺省值barbeat=4,bpm=120,base=0创建SMSP项目

举例: m[4,130,0] （见幻化成风.txt）

然后我们可能要插入几条音轨。先来看语法

=>语言点：音轨控制符

语法：(文件开头或至少一个留白字符)[音轨名]

解析：之所以要留白是为了与方法控制符m[…] 以及其它形如function[…]的函数或控制符号做区分。这个控制符的功能是创建或切换音轨。如果音轨名对应的音轨从未被创建过，则创建一个以它命名的音轨，否则切换到这个音轨。如果开头从未指定过音轨（比如helloworld.txt），则默认创建一个未命名音轨（相当于输入[],中括号里无音轨名称）为缺省音轨。

举例: [P] （见幻化成风.txt）

下面具体讲解音轨这个概念。SMSP中的音轨与DAW软件中的音轨极其类似，作用都是提供一系列在时间、空间和功能上独立并行的音乐环境。每个音轨可被当作一个对象，拥有不同的参数属性。最主要的属性莫过于音量和乐器。

=>语言点：音色控制符

语法：i[乐器名]

解析：将当前音轨的乐器设置为乐器名字符串对应的乐器。这里的乐器可以是MIDI乐器，也可以是直接渲染出波形的广义乐器。如果从未指定，则音轨默认的缺省乐器为MIDI钢琴。详见关于编译器的说明文档。

举例：i[Midi[Piano]] （见幻化成风.txt）

=>语言点：音量控制符

语法：v[音量]

解析：将当前音轨的音量设置为音量字符串对应的正实数，以振幅大小为单位，0表示无声音(-∞dB)，1表示未调整音量(0dB)。如果从未指定，则音轨默认的音量为1

举例：v[0.6] （见幻化成风.txt）

因此我们在导入文件和设置全局变量后写下如下代码：

[音轨1] i[乐器1] v[音量1]

[音轨2] i[乐器2] v[音量2]

…

按照顺序解释的思想，其含义就是创建音轨1，赋予它乐器1和音量1属性，然后创建音轨2，赋予乐器2和音量2属性…以此类推。

以上可被认为是标准的创作音乐前的准备工作。

1. SMSP里的时间轴是什么

继续上面讨论，音轨还拥有时间轴的属性。但不同于DAW软件所有音轨公用一个时间轴的情况，在SMSP里，每个音轨单独分配时间轴，缺省时它们均位于时间原点

在每个音轨内部，有3个基本的控制时间轴的符号。

=>语言点：主小节控制符

语法：|

解析：将当前时间轴对应时间根据barbeat和bpm的值增加一个小节，具有强制对齐的属性

举例：1 1 5 5|6 6 5-

=>语言点：跨小节控制符

语法：音符+小节前拍长+)+小节后拍长

解析：与|类似，但是无强制对齐和中断渲染的属性，专门用于表示跨小节的音符

举例：1- 2-)- 3--|中，音符2共3拍，切分后，小节前2拍，小节后1拍

若连续跨多个小节，可以连写)符号，比如1- 2-)----)-- 3-中这个2连续跨了2个小节

=>语言点：小节控制符

语法：(小节数)

解析：设置并强制对齐音轨的时间轴为某一小节开始处。小节数为0时默认为时间原点，通常是正整数，当然为了自由度也可以是正实数甚至是负数（时间<0的部分会被忽略掉）

举例：(0) 1 1 5 5|6 6 5-

下面介绍一下强制对齐机制。

在每个小节都是饱满的情况下，强制对齐的作用不明显，比如以下3种方式等价：

1 1 5 5|6 6 5-|4 4 3 3|2 2 1-

⬄

1 1 5 5 6 6 5- 4 4 3 3 2 2 1-

⬄

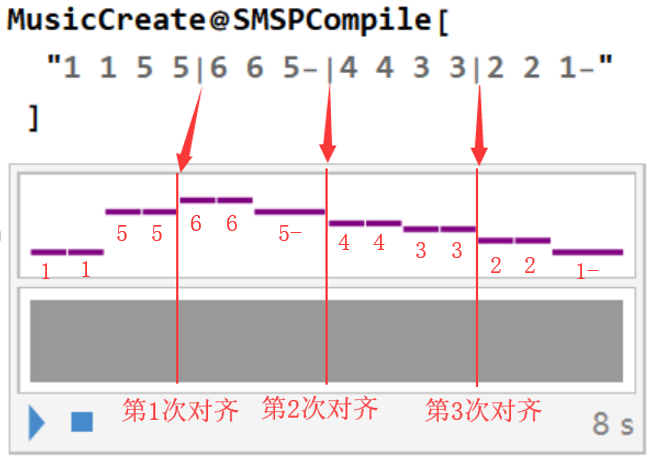
(0) 1 1 5 5

(1) 6 6 5

(2) 4 4 3 3

(3) 2 2 1

⬄（钢琴卷帘表示）

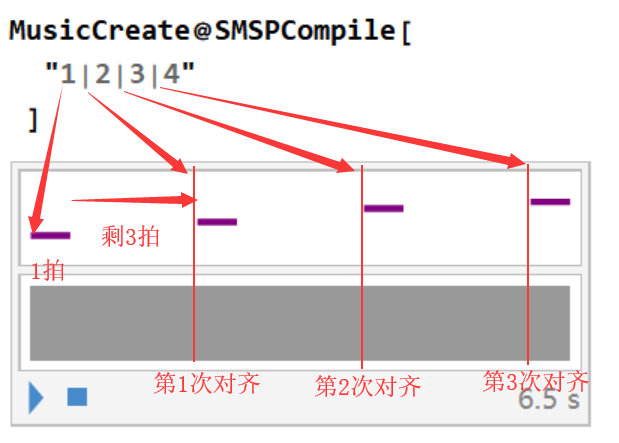


当每小节不满时，强制对齐会导致区别，看以下2组等价关系的区别：

第一组是

1|2|3|4 ⬄ 1 0--|2 0--|3 0--|4 0-- ⬄ 1 0-- 2 0-- 3 0-- 4 0-- ⬄(0) 1 (1) 2 (2) 3 (3) 4

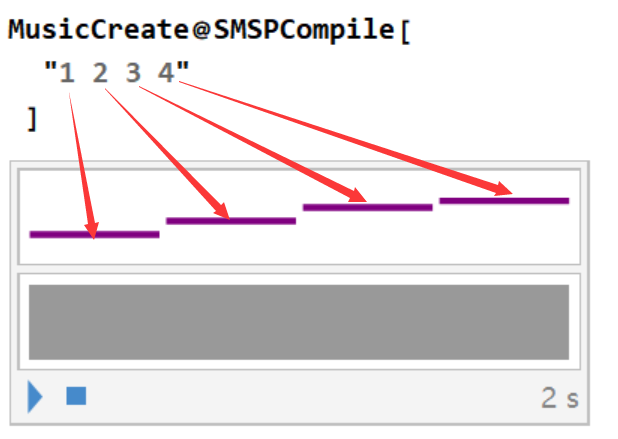
⬄（钢琴卷帘表示）



第二组是

1 2 3 4 ⬄ (0) 1 2 3 4

⬄（钢琴卷帘表示）



可见1|2|3|4与1 2 3 4是不等价的，小节不满时，强制对齐等价于在小节结尾补上休止符。

当每小节超满时，强制对齐也会导致区别，看以下的等价关系：

1 1 5 5 6 6 5-|4 4 3 3|2 2 1-

⬄

(0) 1 1 5 5 6 6 5-

(1) 4 4 3 3

(2) 2 2 1

⬄

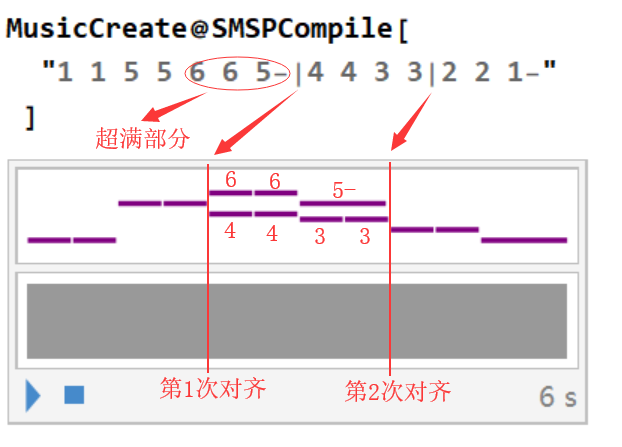
(0) 1 1 5 5

(1) 6 6 5-

(1) 4 4 3 3

(2) 2 2 1

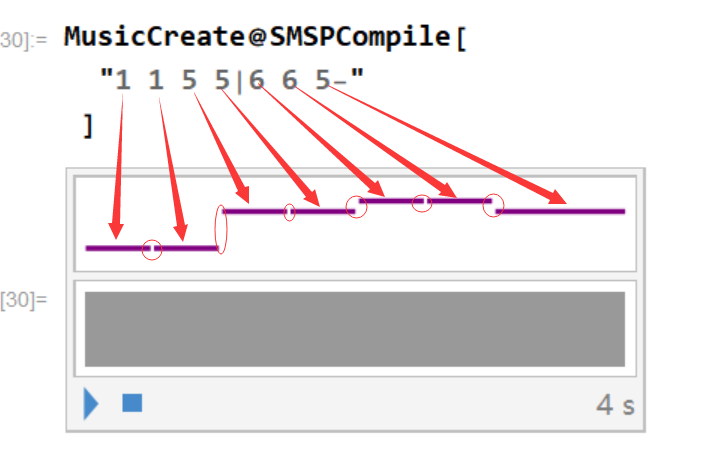
⬄（钢琴卷帘表示）



可见中间漏了一个主小节控制符|后，如上面红色所示会出现音符重叠的情况。

1. 什么是节拍与延音

对于仅包含节拍的乐谱记法来说，一个音符的结束对应下一个音符的开始，举字母歌前2小节为例：



1 1 5 5|6 6 5- 共有7个音符，如箭头所示。有6个连接处，如圆圈所示。

也就是说，在同一音轨内，如果未触发强制对齐机制，节拍体现了该音符与下一音符的开始时间差，并且默认该音符的持续时间与节拍对应时间相等。

但是对于实际的音乐来说，音符是可能重叠的。所以我们需要额外引入延音的概念，表示音符持续时间与节拍对应时间不相等的情况。

总结一下用公式表示就是：

节拍= (下一音符开始时间-该音符开始时间)\*(60/bpm)

延音= 该音符结束时间-该音符开始时间

默认情况：下一音符开始时间=该音符结束时间，即延音=节拍\*(60/bpm)

=>语言点: 音符的节拍表示

语法：

1. 节拍字符串写在音名的后面，必须为特殊符号序列
2. 按照加法规则计算总节拍
3. 允许用任意多的)表示跨小节
4. 允许空缺且支持自定义符号

解析：

SMSP默认的节拍符号如下所示：



当空缺时，表示1拍。

不空缺时，除了”-”其余符号对应数字加起来计算总拍长，不计顺序。

”-”代表的2拍实际上应理解为”+1拍”，第1个符号是”-”也按空缺处理

举例：

x => 1拍（空缺）

x=\_ => 1/4+1/2=3/4拍

x-- =>初始1拍（空缺），总共1+1+1=3拍

x.- =>3/2+1=5/2拍

x-= =>初始1拍（空缺），总共1+1+1/4=9/4拍

对于非默认情况，即延音节拍\*(60/bpm)时，出现音符重叠情况，在SMSP中有两种可选择但不兼容的处理方式。一种是定长延音，一种是非定长延音。

首先介绍定长延音，顾名思义，它表示延音的时长是个定值，下面看语法

=>语言点：音符的定长延音

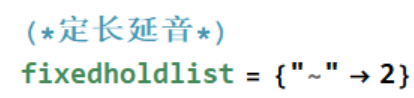
语法：(1) 定长延音字符串写在节拍字符串的后面，必须为特殊符号序列

(2) 按照加法规则计算总延音

(3) 允许空缺且支持自定义符号

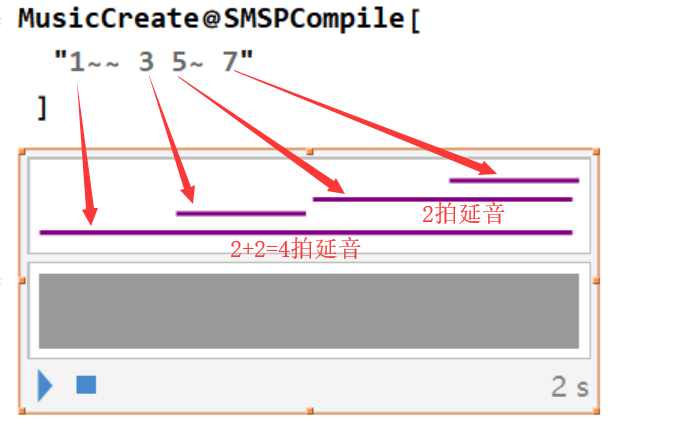
解析：

SMSP默认的定长延音符号如下所示（目前只有1种）：



也就是说根据加法规则音符的总延音就是“2倍~的数量\*1拍”，需要注意的是，延音的长度若小于节拍的对应时长或空缺，回到默认延音=节拍\*(60/bpm)的情况

举例：



然后是非定长延音，指的是不能立刻确定延音